```
1: // $Id: 00-trivial.oc,v 1.4 2018-03-30 10:19:09-07 - - $
2: //
3: // This program does nothing but does not produce error messages.
4: //
```

```
1: // $Id: 01-hello.oc,v 1.4 2018-03-30 10:19:09-07 - - $
2: // Simple hello world program.
3:
4: #include "oclib.oh"
5:
6: void main() {
7:    puts ("Hello, world!\n");
8: }
9:
```

```
 1: // $Id: 03-test3.oc,v 1.5 2018-03-30 10:19:09-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: void main() {
 6:    int a = 3;
 7:    int b = 8;
 8:    int c = a + b;
 9:    a = b + c;
10:    puti (a);
11:    putc ('\n');
12: }
13:
```

```
 1: // $Id: 04-test4.oc,v 1.5 2018-03-30 10:21:02-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: struct foo {
 6:    int a;
 7: }
 8:
 9: void main() {
10:    int a = 6;
11:    foo b = new foo();
12:    b->a = 8;
13:    a = a * b->a + 6;;
14:    puti (a);
15:    putc (' ');
16:    puti (b->a);
17:    endl();
18: }
19:
```

```
 1: // $Id: 06-test6.oc,v 1.6 2018-03-30 10:19:09-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: struct foo {}
 6: struct bar {}
 7:
 8: int f0();
 9: int f1 (int a);
10: int f2 (int a, int b);
11: int f3 (string a, string b, string c);
12: int f4 (foo a, bar b);
13:
14: void main() {
15:    string s = "a";
16:    string[] sa = new string[10];
17: }
18:
```

```
1: // $Id: 07-assert.oc,v 1.5 2018-03-30 10:19:09-07 - - $
2:
3: #include "oclib.oh"
4:
5: void main() {
6:    assert ("null" == null);
7: }
8:
```

```
 1: // $Id: 10-hundred.oc,v 1.4 2018-03-30 10:19:09-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: void main() {
 6:    int count = 0;
 7:    while (count <= 100) {
 8:       count = count + 1;
 9:       puti (count);
10:       endl();
11:    }
12: }
```

```
 1: // $Id: 11-numbers.oc,v 1.4 2018-03-30 10:19:09-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: void main() {
 6:    int number = 1;
 7:    while (number > 0) {
 8:       puti (number);
 9:       putc ('\n');
10:       number = number + number;
11:    }
12:    puti (number);
13:    putc ('\n');
14: }
15:
```

```
 1: // $Id: 12-elseif.oc,v 1.5 2018-03-30 10:19:09-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: int a = 3;
 6:
 7: void main() {
 8:    if (a == 1) puts ("one");
 9:    else if (a == 2) puts ("two");
10:    else if (a == 3) puts ("three");
11:    else puts ("many");
12:    endl();
13: }
14:
```

```
 1: // $Id: 13-assertfail.oc,v 1.6 2018-03-30 10:19:09-07 - - $
 2:
 3: #undef __OCLIB_OH__
 4: #include "oclib.oh"
 5:
 6: void main (int argc, string[] argv) {
 7:    puts (argv[0]);
 8:    puts (" was compiled on ");
 9:    puts (__DATE__);
10:    puts (" @ ");
11:    puts (__TIME__);
12:    endl();
13: }
14:
```

```
 1: // $Id: 14-ocecho.oc,v 1.5 2018-03-30 10:19:09-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: void main(int argc, string[] argv) {
 6:    int argi = 1;
 7:    while (argi < argc) {
 8:       if (argi > 1) putc (' ');
 9:       puts (argv[argi]);
10:       argi = argi + 1;
11:    }
12:    endl();
13: }
14:
```

```
 1: // $Id: 20-fib-array.oc,v 1.6 2018-03-30 10:19:09-07 - - $
 2: //
 3: // Put Fibonacci numbers in an array, then print them.
 4: //
 5:
 6: #include "oclib.oh"
 7:
 8: #define FIB_SIZE 30
 9:
10: void main() {
11:    int[] fibonacci = new int[FIB_SIZE];
12:    fibonacci[0] = 0;
13:    fibonacci[1] = 1;
14:
15:    int index = 2;
16:    while (index < FIB_SIZE) {
17:       fibonacci[index] = fibonacci[index - 1] + fibonacci[index - 2];
18:       index = index + 1;
19:    }
20:
21:    index = 0;
22:    puts ("Numeri di figlio Bonacci\n");
23:    while (index < FIB_SIZE) {
24:       puts ("fibonacci[");
25:       puti (index);
26:       puts ("] = ");
27:       puti (fibonacci[index]);
28:       endl();
29:       index = index + 1;
30:    }
31: }
32:
```

```
 1: // $Id: 21-eratosthenes.oc,v 1.6 2018-03-30 10:19:09-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: #define SIZE 100
 6: #define LOWPRIME 2
 7:
 8: void main() {
 9:    bool[] sieve = new bool[SIZE];
10:    int index = LOWPRIME;
11:
12:    while (index < SIZE) {
13:       sieve[index] = true;
14:       index = index + 1;
15:    }
16:
17:    int prime = LOWPRIME;
18:    while (prime < SIZE) {
19:       if (sieve[prime]) {
20:          index = prime * 2;
21:          while (index < SIZE) {
22:             sieve[index] = false;
23:             index = index + prime;
24:          }
25:       }
26:       prime = prime + 1;
27:    }
28:
29:    index = LOWPRIME;
30:    while (index < SIZE) {
31:       if (sieve[index]) {
32:          puti (index);
33:          endl();
34:       }
35:       index = index + 1;
36:    }
37: }
38:
```

```
 1: // $Id: 23-atoi.oc,v 1.13 2018-03-30 10:19:09-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: int atoi (string str) {
 6:    assert (str != null);
 7:    bool neg = false;
 8:    int num = 0;
 9:    int digit = 0;
10:    if (str[0] != '\0') {
11:       if (str[0] == '-') {
12:          digit = digit + 1;
13:          neg = true;
14:       }
15:       bool contin = true;
16:       while (contin) {
17:          if (str[digit] == '\0') {
18:             contin = false;
19:          }else {
20:             char c = str[digit];
21:             digit = digit + 1;
22:             if (c < '0') contin = false;
23:             else if (c > '9') contin = false;
24:             else num = num * 10 + c - '0';
25:          }
26:       }
27:       if (neg) num = - num;
28:    }
29:    return num;
30: }
31:
32: void main (int argc, string[] argv) {
33:    int argi = 1;
34:    while (argi < argc) {
35:       string arg = argv[argi];
36:       puts (arg);
37:       puts (" = ");
38:       puti (atoi (arg));
39:       endl();
40:       argi = argi + 1;
41:    }
42: }
43:
```

```
 1: // $Id: 30-fac-fnloop.oc,v 1.7 2018-03-30 10:19:09-07 - - $
 2: //
 3: // Function uses a loop to compute factorial.
 4: //
 5:
 6: #include "oclib.oh"
 7:
 8: int fac (int n) {
 9:    int f = 1;
10:    while (n > 1) {
11:       f = f * n;
12:       n = n - 1;
13:    }
14:    return f;
15: }
16:
17: void main() {
18:    int n = 1;
19:    while (n <= 5) {
20:       puti (fac (n));
21:       endl();
22:       n = n + 1;
23:    }
24: }
25:
```

```
 1: // $Id: 31-fib-2supn.oc,v 1.6 2018-03-30 10:21:02-07 - - $
 2: //
 3: // Very slow program, computes Fibonacci numbers with O(2^n) speed.
 4: //
 5:
 6: #include "oclib.oh"
 7:
 8: int fibonacci (int n) {
 9:    if (n < 2) return n;
10:    return fibonacci (n - 1) + fibonacci (n - 2);
11: }
12:
13: void main() {
14:    int n = 0;
15:    while (n < 10) {
16:       puts ("fibonacci(");
17:       puti (n);
18:       puts (") = ");
19:       puti (fibonacci (n));
20:       endl();
21:       n = n + 1;
22:    }
23: }
24:
```

```
 1: // $Id: 40-arraystack.oc,v 1.11 2018-03-30 11:08:23-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: #define EMPTY (-1)
 6:
 7: struct stack {
 8:    string[] data;
 9:    int size;
10:    int top;
11: }
12:
13: stack new_stack (int size) {
14:    stack stack = new stack(); // Zeros out both fields
15:    stack->data = new string[size]; // Array of null pointers
16:    stack->size = size;
17:    stack->top = EMPTY;
18:    return stack;
19: }
20:
21: void push (stack stack, string str) {
22:    assert (stack->top < stack->size - 1);
23:    stack->top = stack->top + 1;
24:    stack->data[stack->top] = str;
25: }
26:
27: string pop (stack stack) {
28:    assert (stack->top != EMPTY);
29:    string tmp = stack->data[stack->top];
30:    stack->top = stack->top - 1;
31:    return tmp;
32: }
33:
34: bool empty (stack stack) {
35:    return stack->top == EMPTY;
36: }
37:
38: void main (int argc, string[] argv) {
39:    stack stack = new_stack (100);
40:
41:    int argi = 0;
42:    while (argi < argc) {
43:       push (stack, argv[argi]);
44:       argi = argi + 1;
45:    }
46:
47:    while (! empty (stack)) {
48:       puts (pop (stack));
49:       endl();
50:    }
51: }
52:
```

```
 1: // $Id: 41-linkedstack.oc,v 1.12 2018-03-30 11:08:23-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: struct node {
 6:    string data;
 7:    node link;
 8: }
 9:
10: struct stack {
11:    node top;
12: }
13:
14: bool empty (stack stack) {
15:    assert (stack != null);
16:    return stack->top == null;
17: }
18:
19: stack new_stack() {
20:    stack stack = new stack();
21:    stack->top = null;
22:    return stack;
23: }
24:
25: void push (stack stack, string str) {
26:    assert (stack != null);
27:    node tmp = new node();
28:    tmp->data = str;
29:    tmp->link = stack->top;
30:    stack->top = tmp;
31: }
32:
33: string pop (stack stack) {
34:    assert (stack != null);
35:    assert (! empty (stack));
36:    string tmp = stack->top->data;
37:    stack->top = stack->top->link;
38:    return tmp;
39: }
40:
41: void main (int argc, string[] argv) {
42:    stack stack = new_stack();
43:    int argi = 0;
44:
45:    while (argi < argc) {
46:       push (stack, argv[argi]);
47:       argi = argi + 1;
48:    }
49:
50:    while (! empty (stack)) {
51:       puts (pop (stack));
52:       endl();
53:    }
54: }
55:
```

```
 1: // $Id: 42-viiiqueens.oc,v 1.9 2018-03-30 10:21:02-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: #define BOARD_SIZE 8
 6: int[] board = new int[BOARD_SIZE];
 7:
 8: bool is_safe (int newcol) {
 9:    int col = 0;
10:    while (col < newcol) {
11:       if (board[col] == board[newcol]) return false;
12:       int diagonal = board[col] - board[newcol];
13:       if (diagonal == col - newcol) return false;
14:       if (diagonal == newcol - col) return false;
15:       col = col + 1;
16:    }
17:    return true;
18: }
19:
20: void printqueens() {
21:    int col = 0;
22:    while (col < BOARD_SIZE) {
23:       putc (board[col] + '1');
24:       col = col + 1;
25:    }
26:    putc ('\n');
27: }
28:
29: void queens (int newcol) {
30:    if (newcol == BOARD_SIZE) printqueens();
31:    else {
32:       int row = 0;
33:       while (row < BOARD_SIZE) {
34:          board[newcol] = row;
35:          if (is_safe (newcol)) queens (newcol + 1);
36:          row = row + 1;
37:       }
38:    }
39: }
40:
41: void main() {
42:    queens (0);
43: }
44:
```

```
 1: // $Id: 44-dot-product.oc,v 1.8 2018-03-30 10:21:02-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: int dot_product (int size, int[] vec1, int[] vec2) {
 6:    int index = 0;
 7:    int dot = 0;
 8:    while (index < size) {
 9:       dot = dot + vec1[index] * vec2[index];
10:       index = index + 1;
11:    }
12:    return dot;
13: }
14:
15: #define SIZE 10
16:
17: int[] vec1 = new int[SIZE];
18: int[] vec2 = new int[SIZE];
19:
20: void main() {
21:    int i = 0;
22:    while (i < SIZE) {
23:       vec1[i] = i + 10;
24:       vec2[i] = i * 10;
25:       i = i + 1;
26:    }
27:    puti (dot_product (SIZE, vec1, vec2));
28:    endl();
29: }
30:
```

```
 1: // $Id: 45-towers-of-hanoi.oc,v 1.6 2018-03-30 10:19:09-07 - - $
 2:
 3: #include "oclib.oh"
 4:
 5: void move (string src, string dst) {
 6:    puts ("Move a disk from ");
 7:    puts (src);
 8:    puts (" to ");
 9:    puts (dst);
10:    puts (".\n");
11: }
12:
13: void towers (int ndisks, string src, string tmp, string dst) {
14:    if (ndisks < 1) return;
15:    towers (ndisks - 1, src, dst, tmp);
16:    move (src, dst);
17:    towers (ndisks - 1, tmp, src, dst);
18: }
19:
20: void main() {
21:    towers (4, "Source", "Temporary", "Destination");
22: }
23:
```

```
 1: // $Id: 53-insertionsort.oc,v 1.10 2018-03-30 10:21:02-07 - - $
 2: //
 3: // Use insertion sort to print argv in sorted order.
 4: //
 5:
 6: #include "oclib.oh"
 7:
 8: int strcmp (string s1, string s2) {
 9:    int index = 0;
10:    bool contin = true;
11:    while (contin) {
12:       char s1c = s1[index];
13:       char s2c = s2[index];
14:       int cmp = s1c - s2c;
15:       if (cmp != 0) return cmp;
16:       if (s1c == '\0') contin = false;
17:       index = index + 1;
18:    }
19:    return 0;
20: }
21:
22: void insertion_sort (int size, string[] array) {
23:    int sorted = 1;
24:    while (sorted < size) {
25:       int slot = sorted;
26:       string element = array[slot];
27:       bool contin = true;
28:       while (contin) {
29:          if (slot == 0) {
30:             contin = false;
31:          }else if (strcmp (array[slot - 1], element) <= 0) {
32:             contin = false;
33:          }else {
34:             array[slot] = array[slot - 1];
35:             slot = slot - 1;
36:          }
37:       }
38:       array[slot] = element;
39:       sorted = sorted + 1;
40:    }
41: }
42:
43: void print_array (string label, int size, string[] array) {
44:    endl();
45:    puts (label);
46:    puts (":\n");
47:    int index = 0;
48:    while (index < size) {
49:       puts (array[index]);
50:       endl();
51:       index = index + 1;
52:    }
53: }
54:
55: void main (int argc, string[] argv) {
56:    print_array ("unsorted", argc, argv);
57:    insertion_sort (argc, argv);
58:    print_array ("sorted", argc, argv);
```

```
59: }
60:
```

```
1: char O[9];Q(l,b,d){int o=8,p=1,q=1<<
2: l|1<<22-l;for(;l>7?!write(1,O,9):o--
3: ;)O[l]=56-o,b&p|d&q||Q(l+1,b|p,d|q),
4: p*=2,q*=2;}main(){O[8]=10;Q(0,0,0);}
```

```
 1: // $Id: 91-typecheck.oc,v 1.5 2018-03-30 10:21:02-07 - - $
 2: //
 3: // This file should scan and parse correctly,
 4: // but fail to type check.
 5: //
 6:
 7: int[] a = null;
 8: reference[] a = new string[10];
 9: void foo();
10: void foo (int a);
11: void foo (int[] a, int[] b) {int x = a + b;}
12: struct foo { int a; int b; }
13:
14: void main() {
15:    a + b;
16:    f();
17:    f (x, y+3, z);
18:    foo + bar;
19:    a = b = c = d;
20:    test = abc + def + ghi;
21:    this + 23 * a + "hello";
22:    while (a < b) f = f + 1;
23:    return 3 + 4;
24:    a[i] = b[j];
25:    return;
26:    while (true) {a = 3; b = 4; }
27:    if (a == b) f (x);
28:    if (a != b) y = 3; else f (y, z);
29: }
30:
```

```
 1: /*
 2: This is an unterminated comment.
 3: It would cause cpp to error out.
 4: When cpp returns a non-zero exit code,
 5: so should your compiler.
 6: $Id: 92-uncomment.oc,v 1.3 2018-03-30 10:19:09-07 - - $
 7:
 8: int main (int argc, char **argv) {
 9:
10:    Your compiler never sees any of this code.
11:
12: }
```

```
 1: // $Id: 93-semantics.oc,v 1.4 2018-03-30 10:19:09-07 - - $
 2: // This code should scan and parse correctly,
 3: // but fail to type check.
 4: int[] a = null;
 5: int[] b = null;
 6:
 7: void main() {
 8:    int c = a + b; // can't add arrays
 9:    void[] f() {}; // can't hae void[]
10:    void n = null; // can't have void vars
11:    bool x = a < b; // can't compare pointers <
12:    bool y = a==b; // this is ok
13: }
14:
```

```
 1: // $Id: 94-syntax.oc,v 1.3 2018-03-30 10:19:09-07 - - $
 2:
 3: k
 4: int f() {
 5: int a = ;
 6: return foo;
 7: public static void main (String[] args) {
 8:    System.exit (255);
 9: }
10:
```

```
 1: // $Id: 95-cobol.oc,v 1.3 2018-03-30 10:19:09-07 - - $
 2:
 3: 000100 IDENTIFICATION DIVISION.
 4: 000200 PROGRAM-ID.     HELLOWORLD.
 5: 000300
 6: 000400*
 7: 000500 ENVIRONMENT DIVISION.
 8: 000600 CONFIGURATION SECTION.
 9: 000700 SOURCE-COMPUTER. RM-COBOL.
10: 000800 OBJECT-COMPUTER. RM-COBOL.
11: 000900
12: 001000 DATA DIVISION.
13: 001100 FILE SECTION.
14: 001200
15: 100000 PROCEDURE DIVISION.
16: 100100
17: 100200 MAIN-LOGIC SECTION.
18: 100300 BEGIN.
19: 100400     DISPLAY " " LINE 1 POSITION 1 ERASE EOS.
20: 100500     DISPLAY "Hello world!" LINE 15 POSITION 10.
21: 100600     STOP RUN.
22: 100700 MAIN-LOGIC-EXIT.
23: 100800     EXIT.
```

```
 1: // Unterminated strings.
 2: // $Id: 96-unterminated.oc,v 1.5 2018-03-30 10:19:09-07 - - $
 3:
 4: void main() {
 5:    string t = "\*/";
 6:    string s = "abc;
 7:    char c = 'a;
 8:    s = "abcd\";
 9:    s = "abc|\
10:    ;
11:    int 23foobar;
12: }
13:
```